Cryptography I, homework sheet 6

Due: 16 October 2014, 10:45

Team up in groups of two or three to hand in your homework. We do not have capacity to correct all homeworks individually. To submit your homework, email it to `crypto14@tue.nl` or place it on the lecturer's table before the lecture. Do not email Tanja or put homework in mailboxes. You may use computer algebra systems such as mathematica and sage; please submit your code as part of your homework.

1. Majordomo is a program that manages Internet mailing lists. If you send a message to `majordomo@foodplus.com` saying `subscribe recipes`, Majordomo will add you to the `recipes` mailing list, and you will receive several interesting recipes by e-mail every day.

   It is easy to forge mail. You can subscribe a victim, let's say `God@heaven.af.mil`, to the `recipes` mailing list, and thousands more mailing lists, by sending fake subscription requests to Majordomo. `God@heaven.af.mil` will then be flooded with mail.

   Majordomo 1.94, released in October 1996, attempts to protect subscribers as follows. After it receives your subscription request, it sends you a confirmation number. To complete your subscription, you must send a second request containing the confirmation number.

   Majordomo 1.94 generates confirmation numbers as follows. There is a function h that changes strings to numbers. The `recipes` mailing list has a secret string k. The confirmation number for an address a is $h(ka)$. For example, if the secret string is `ossifrage`, and the address is `God@heaven.af.mil`, the confirmation number is $h($`ossifrageGod@heaven.af.mil`$)$.

   The function h produces a 32-bit result, computed as follows. Start with 0. Add the first byte of the string. Rotate left 4 bits. Add the next byte of the string. Rotate left 4 bits. Continue adding and rotating until the end of the string.

   Explain how to subscribe `God@heaven.af.mil` to the `recipes` mailing list despite this protection, and explain what Majordomo 1.94 should have done.

2. Message authentication codes can be built from ingredients other than keyed hash functions. Wegman and Carter proposed an approach that is nformation-theoretically secure, i.e., there is a guaranteed limit on the attacker's chance of success no matter how much computation the attacker does. The following only explains how to authenticate a fixed number of messages, see `http://cr.yp.to/mac.html` for full details on a deployable system.

   Here is a toy version with which A and B can authenticate t messages: Fix a prime p, e.g. $p = 1000003$. Randomly generate integers $r, s_1, s_2, \dots, s_t \in \{0, 1, 2, \dots 1000002\}$. These values are the shared secrets; r is the overall secret and the s_i are per message secrets.

   To authenticate the i-th message m_i the sender expresses m_i in base p as $m_i = m_{i,0} + m_{i,1}p + m_{i,2}p^2 + ... + m_{i,n}p^n$ and computes the authenticator as

   $$a = m_{i,0}r + m_{i,1}r^2 + m_{i,2}r^3 + ... + m_{i,n}r^{n+1} + s_i \bmod p.$$

   For simplicity we will do $i = 1$ and omit the extra indices. Compute the authenticator for $m = 454356542435979283475928437, r = 483754, s = 342534$.